**SOMMAIRE**

[INTRODUCTION 3](#_Toc138016335)

[I. ARCHITECTURE DU PROJET 3](#_Toc138016336)

[I.1. Architecture globale 3](#_Toc138016337)

[I.2. Frontend 5](#_Toc138016338)

[I.3. Backend 6](#_Toc138016339)

[I.4. Base de données 7](#_Toc138016340)

[II. PRESENTATION DES TECHNOLOGIES UTILISEES 7](#_Toc138016341)

[II.1. Le backend 7](#_Toc138016342)

[II.2. Le frontend React 10](#_Toc138016343)

[II.3. La base de données 15](#_Toc138016344)

[III. MISE EN PLACE DE L’ENVIRONNEMENT DE TRAVAIL 16](#_Toc138016345)

[III.1. Processus d’installation des Framework 16](#_Toc138016346)

[IV. REALISATION 22](#_Toc138016347)

[IV.1. Arborescence globale du projet 22](#_Toc138016348)

[IV.2. Présentation des interfaces 27](#_Toc138016349)

**LISTE DES FIGURES**

[**Figure 1 : Architecture globale du projet** 4](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016350)

[**Figure 2 : Architecture du frontend** 6](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016351)

[**Figure 3 : Architecture du backend** 6](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016352)

[**Figure 4 : Logo du langage Python** 8](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016353)

[**Figure 5 : Logo de Pycharm** 8](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016354)

[**Figure 6 : Logo de Postman** 9](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016355)

[**Figure 7 : Logo de Flask** 10](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016356)

[**Figure 8 : Logo de JSON** 10](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016357)

[**Figure 9 : Logo de React** 12](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016358)

[**Figure 10 : Logo de Javascript** 12](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016359)

[**Figure 11 : Logo de Node.js** 13](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016360)

[**Figure 12 : Logo de npm** 14](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016361)

[**Figure 13 : Logo de React-Bootstrap** 15](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016362)

[**Figure 14 : Logo de Visual Studio Code** 15](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016363)

[**Figure 15 : Logo de Apache** 16](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016364)

[**Figure 16 : Logo de MySQL** 16](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016365)

[**Figure 17 : Affichage par défaut de notre application React** 18](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016366)

[**Figure 18 : Arborescence du backend** 22](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016367)

[**Figure 19 : Arborescence principale du frontend** 24](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016368)

[**Figure 20 : Le dossier node\_modules** 24](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016369)

[**Figure 21 : Le dossier public** 25](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016370)

[Figure 22 : Le dossier src 26](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016371)

[**Figure 23 : Site principal (partie 1)** 27](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016372)

[**Figure 24 : Fonction Javascript permettant d'afficher la partie 1 du site principal** 28](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016373)

[**Figure 25 : Code Typescript (HTML) permettant l'affichage de la partie 1 du site** 28](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016374)

[**Figure 26 : Fonction Python permettant l'affichage des publicités** 28](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016375)

[**Figure 27 : Site principal (partie 2)** 29](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016376)

[**Figure 28 : Site principal (partie 3)** 29](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016377)

[**Figure 29 : Code Javascript permettant l'affichage des évènements** 30](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016378)

[**Figure 30 : Code Typescript permettant l'affichage des évènements** 31](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016379)

[**Figure 31 : Code Python permettant l'affichage des évènements** 31](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016380)

[**Figure 32 : Interface de connexion** 31](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016381)

[**Figure 33 : Code Javascript permettant la connexion** 31](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016382)

[**Figure 34 : Code Typescript pour le formulaire de connexion** 32](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016383)

[**Figure 35 : Code Python permettant la connexion** 32](https://d.docs.live.net/b9ec43885d0afd50/Documents/GitHub/ProjetTO52/Documents/DOCUMENT%20TECHNIQUE%20version%203.docx#_Toc138016384)

## ARCHITECTURE DU PROJET

### Architecture globale

L’architecture globale de notre projet se décline comme suit :

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

Figure 1 : Architecture globale du projet

L’architecture globale de notre système se décline en trois principales parties :

* Le frontend est basé sur la bibliothèque React.js, qui est un Framework JavaScript populaire pour la construction d'interfaces utilisateur. Il s'agit d'une bibliothèque JavaScript open-source qui facilite la création d'interfaces utilisateur réactives et interactives. Il utilise également le serveur d’applications NodeJS ainsi que le gestionnaire de paquets npm. Pour envoyer des requêtes HTTP depuis le frontend React.js, le code utilise généralement la fonction **fetch**, qui est une API JavaScript intégrée aux navigateurs modernes. **fetch** permet d'effectuer des requêtes HTTP vers des ressources distantes et de traiter les réponses. Cette API prend en charge les méthodes HTTP telles que GET, POST, PUT, DELETE ainsi que la gestion des en-têtes et des données. Lorsqu’une action est déclenchée dans l’interface utilisateur, par exemple un clic sur un bouton, le code JavaScript dans le frontend utilise fetch pour envoyer une requête http au backend. La requête peut contenir des données supplémentaires, telles que des paramètres de requête ou des données de formulaire, qui sont encodées au format JSON. Le backend traite ensuite la requête, effectue les opérations nécessaires et renvoie une réponse, qui est reçue par le frontend via fetch au format JSON.
* Notre backend est basé sur le Framework Flask. Il permet de gérer les requêtes HTTP, traiter les données et renvoyer des réponses au frontend. Dans Flask, nous avons défini des routes pour spécifier les URL auxquelles notre API répondra. Lorsque le frontend envoie une requête HTTP à notre backend Flask, la route correspondante est déclenchée. Nous avec spécifié pour chaque route le type de requête et dans la fonction de vue correspondante, il est possible d’accéder aux données de la requête, tels que les paramètres de requête, les données de formulaire, les en-têtes.

Avant de pouvoir interagir avec la base de données MySQL, nous avons établi une connexion grâce au package mysql.connector de Python.

* Une fois la connexion établie entre la base de données et le backend, nous exécutons des requêtes SQL sur la base de données. Cela inclut des opérations telles que la sélection (SELECT), l'insertion (INSERT), la mise à jour (UPDATE), la suppression (DELETE).

### Frontend

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

Figure 2 : Architecture du frontend

Le frontend est basé sur la bibliothèque React.js, qui est un Framework JavaScript populaire pour la construction d'interfaces utilisateur. Il s'agit d'une bibliothèque JavaScript open-source qui facilite la création d'interfaces utilisateur réactives et interactives. Il utilise également le serveur d’applications NodeJS ainsi que le gestionnaire de paquets npm. Pour envoyer des requêtes HTTP depuis le frontend React.js, le code utilise généralement la fonction fetch, qui est une API JavaScript intégrée aux navigateurs modernes. fetch permet d'effectuer des requêtes HTTP vers des ressources distantes et de traiter les réponses. Cette API prend en charge les méthodes HTTP telles que GET, POST, PUT, DELETE ainsi que la gestion des en-têtes et des données. Lorsqu’une action est déclenchée dans l’interface utilisateur, par exemple un clic sur un bouton, le code JavaScript dans le frontend utilise fetch pour envoyer une requête http au backend. La requête peut contenir des données supplémentaires, telles que des paramètres de requête ou des données de formulaire, qui sont encodées au format JSON. Le backend traite ensuite la requête, effectue les opérations nécessaires et renvoie une réponse, qui est reçue par le frontend via fetch au format JSON.

### Backend

Une image contenant texte, capture d’écran, Police, diagramme

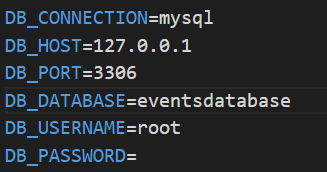
Description générée automatiquement

Figure 3 : Architecture du backend

Notre backend est basé sur le Framework Flask. Il permet de gérer les requêtes HTTP, traiter les données et renvoyer des réponses au frontend. Dans Flask, nous avons défini des routes pour spécifier les URL auxquelles notre API répondra. Lorsque le frontend envoie une requête HTTP à notre backend Flask, la route correspondante est déclenchée. Nous avec spécifié pour chaque route le type de requête et dans la fonction de vue correspondante, il est possible d’accéder aux données de la requête, tels que les paramètres de requête, les données de formulaire, les en-têtes.

### Base de données

Notre base de données est une base MySQL, avec 9 tables. Les paramètres de connexion à notre base de données sont :



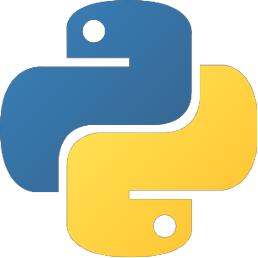
## PRESENTATION DES TECHNOLOGIES UTILISEES

### II.1. Le backend

#### II.1.1. Le langage de programmation Python

C’est aujourd’hui le langage de référence en programmation. Pour le développement de notre API nous avons opté pour l’utilisation du langage de programmation python car c’est celui qui nous a été imposé pour la réalisation de ce projet. La version de Python utilisée est la version 3.11

Figure 4 : Logo du langage Python

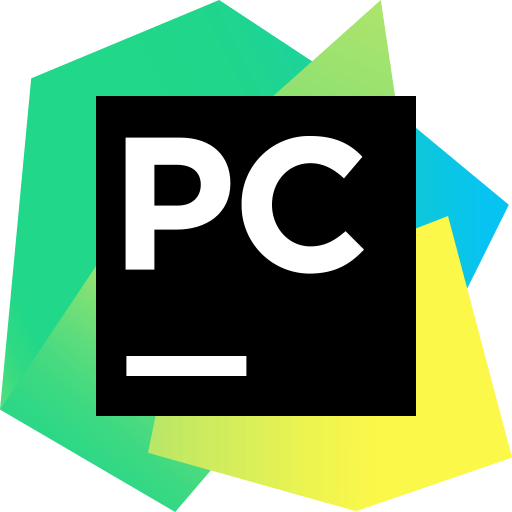


#### II.1.2. Le logiciel Pycharm

PyCharm est un environnement de développement intégré utilisé pour programmer en Python. Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires et l'intégration de logiciel de gestion de versions. Développé par l'entreprise tchèque JetBrains, c'est un logiciel multi-plateforme qui fonctionne sous Windows, Mac OS X et GNU/Linux. Il est décliné en édition professionnelle, diffusé sous licence propriétaire, et en édition communautaire diffusé sous licence Apache.

Nous avons utilisé le logiciel PyCharm pour coder notre API Python.

Figure 5 : Logo de Pycharm



#### II.1.3. Le logiciel Postman

Postman est une application permettant de tester des API, « créée en 2012 par Abhinav Asthana, Ankit Sobti et Abhijit Kane à Bangalore pour répondre à une problématique de test d'API partageable. D'abord module complémentaire de Google Chrome, puis client lourd, et finalement client léger, elle est à présent utilisée par plus de 500 000 entreprises dans le monde et a son siège à San Francisco ». Postman regroupe chaque test d'API dans une collection, permettant de mutualiser leurs URLs et authentifications.

Nous avons utilisé Postman pour réaliser les différents tests sur notre API. La version utilisée est la 9.25.0

Figure 6 : Logo de Postman



#### II.1.4. Le micro Framework Flask

Flask est un micro Framework open-source de développement web en Python. Il est classé comme micro Framework car il est très léger. Flask a pour objectif de garder un noyau simple mais extensible. Il n'intègre pas de système d'authentification, pas de couche d'abstraction de base de données, ni d'outil de validation de formulaires. Cependant, de nombreuses extensions permettent d'ajouter facilement des fonctionnalités. Flask nous a permis de créer l’application Python qui communique avec l’interface en servant de web services.

Une image contenant noir, obscurité

Description générée automatiquement

Figure 7 : Logo de Flask

#### II.1.5. Le format de données JSON

JSON est un format d'échange de données indépendant conçu pour représenter des structures de données simples. JSON est limité aux valeurs textuelles et numériques. Les valeurs binaires ne sont pas supportées. Les structures de données en JSON sont basées sur des paires clé / valeur.

Nous nous sommes basés sur ce format d’échange pour effectuer certaines requêtes d’insertion vers notre base MySQL et pour fournir des services à partir de notre application Python.

Figure 8 : Logo de JSON

Une image contenant cercle, Graphique, capture d’écran, art

Description générée automatiquement

### II.2. Le frontend React

#### II.2.1. La bibliothèque React

React est une bibliothèque JavaScript libre développée par Facebook (maintenant Meta) depuis 2013. Le but principal de cette bibliothèque est de faciliter la création d'application web monopage, via la création de composants dépendant d'un état et générant une page HTML à chaque changement d'état. Nous avons utilisé React pour coder notre frontend. La philosophie de React repose sur quelques principes clés :

* Les composants : React encourage la construction d’applications sous forme de composants réutilisables.
* L’unidirectionnel : les données sont transmises de manière descendante depuis le composant parent vers le composant enfant. Cela facilite le suivi des modifications et rend le code plus prévisible.
* Le Virtual DOM : React utilise un virtual DOM (Document Object Model Virtuel) pour gérer efficacement les mises à jour de l’interface utilisateur et permet d’avoir de bonnes performances.
* La réconciliation : React effectue une réconciliation efficace entre le virtual DOM et le DOM réel pour déterminer les modifications à apporter. Cela permet d’optimiser les mises à jour de l’interface utilisateur en ne modifiant que les parties qui ont réellement changé.
* Javascript : React est conçu pour être utilisé avec Javascript. Il facilite l’intégration du code Javascript existant et permet d’utiliser les fonctionnalités de ce langage pour gérer l’état de l’application et les interactions utilisateur.

Ces principes de base de la philosophie de programmation de React visent à rendre le développement d'interfaces utilisateur plus facile, maintenable et performant, en favorisant la réutilisabilité, la modularité et la gestion efficace des mises à jour de l'interface utilisateur.

Une image contenant Graphique, cercle, art, symbole

Description générée automatiquement

Figure 9 : Logo de React

#### II.2.2. Le langage de programmation Javascript

JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives et à ce titre est une partie essentielle des applications web. Avec les langages HTML et CSS, JavaScript est au cœur des langages utilisés par les développeurs web. Une grande majorité des sites web l'utilisent, et la majorité des navigateurs web disposent d'un moteur JavaScript pour l'interpréter. JavaScript est aussi employé pour les serveurs Web avec l'utilisation de Node.js.

Figure 10 : Logo de Javascript

Une image contenant Police, Graphique, logo, conception

Description générée automatiquement

#### II.2.3. Le serveur d’applications Node.JS

Node.js est une plateforme logicielle libre en JavaScript, orientée vers les applications réseau évènementielles hautement concurrentes qui doivent pouvoir monter en charge. Parmi les modules natifs de Node.js, on retrouve http qui permet le développement de serveur HTTP. Ce qui autorise, lors du déploiement de sites internet et d'applications web développés avec Node.js, de ne pas installer et utiliser des serveurs web tels que Nginx ou Apache.

Concrètement, Node.js est un environnement bas niveau permettant l'exécution de JavaScript côté serveur ce qui fait de Node.js notre serveur d’applications.

Figure 11 : Logo de Node.js

Une image contenant capture d’écran, Graphique, conception

Description générée automatiquement

#### II.2.4. L’outil npm

npm est le gestionnaire de paquets par défaut pour l'environnement d'exécution JavaScript Node.js. npm se compose d'un client en ligne de commande, également appelé npm, et d'une base de données en ligne de paquets publics et privés payants, appelée le registre npm. Le registre est accessible via le client, et les paquets disponibles peuvent être parcourus et recherchés via le site Web de npm. Le gestionnaire de paquets et le registre sont gérés par npm, Inc.

Une image contenant Graphique, rouge, capture d’écran, Police

Description générée automatiquement

Figure 12 : Logo de npm

#### II.2.5. Le package React-Bootstrap

Bootstrap est une collection d'outils utile à la création du design (graphisme, animation et interactions avec la page dans le navigateur, etc.) de sites et d'applications web. Il se base sur un système de grille simple et efficace pour mettre en ordre l'aspect visuel d'une page web. Il apporte du style aux différents composants d’une page web. Nous utilisé Bootstrap v5 à cause de sa simplicité d’utilisation. Il nous a été utile pour le design de notre plateforme web. Le package React-Bootstrap permet d’adapter les composants Bootstrap aux composants React.

Une image contenant symbole, Graphique, logo, Police

Description générée automatiquement

Figure 13 : Logo de React-Bootstrap

#### II.2.6. Le logiciel Visual Studio Code

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et MacOs. Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les snippets, la refactorisation du code et Git intégré.

Nous avons utilisé le logiciel Visual Studio Code pour le code de notre plateforme web d’administration.

Figure 14 : Logo de Visual Studio Code

Une image contenant symbole, Bleu électrique, logo, ligne

Description générée automatiquement

### II.3. La base de données

#### II.3.1. Le serveur web Apache

Apache HTTP Server (Apache) est un serveur HTTP créé et maintenu au sein de la fondation Apache. Jusqu'en avril 2019, ce fut le serveur HTTP le plus populaire du World Wide Web. Il est distribué selon les termes de la licence Apache. La version d’apache utilisée lors de notre développement est la 2.4.53; installée à partir du logiciel XAMPP. Il a permis de déployer en local notre application web d’administration.

Une image contenant stylos et plumes, outil d’écriture, Graphique, conception

Description générée automatiquement

Figure 15 : Logo de Apache

#### II.3.2. Le système de gestion de base de données MySQL

MySQL est un système de gestion de bases de données relationnelles (SGBDR). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde. Il permet de gérer des bases de données relationnelles.

Figure 16 : Logo de MySQL

Une image contenant Graphique, Police, graphisme, logo

Description générée automatiquement

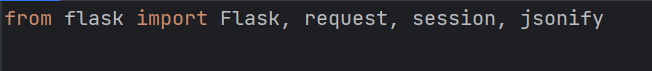
## MISE EN PLACE DE L’ENVIRONNEMENT DE TRAVAIL

### III.1. Processus d’installation des Framework

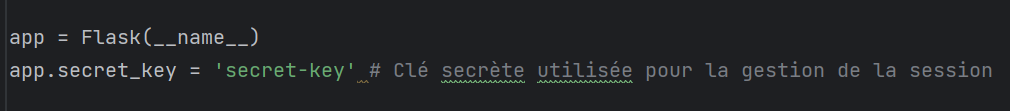
#### III.1.1. Le micro Framework Flask

Le micro Framework Flask se comporte comme un package Python, ce qui simplifie son installation. Il peut s’installer directement dans Pycharm dans l’onglet Python Packages ou dans le terminal python grâce à la commande : $ pip install Flask

Après l’installation, nous importons le package Flask ainsi que les composants dont nous avons besoin grâce à la ligne :



Nous créons ensuite notre application Flask avec la ligne :



Ensuite nous configurons la base de données dans notre application python. Pour ce faire, nous installons le package ***mysql-connector*** qui est le pilote MySQL de Python. Ce package s’installe grâce à la commande : $ pip install mysql-connector

L’importation du package se fait grâce à la ligne : import mysql-connnector

Enfin, nous allons établir la connexion à la base de données grâce à la commande :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Il suffit maintenant de créer nos routes et d’écrire les fonctions correspondantes à ces routes.

#### III.1.2. Le Framework React

Pour décrire le processus d’installation du Framework React, nous avons suivi les instructions de la documentation.

##### Installation de NodeJS

Nous allons d’abord installer NodeJS avec npm. La version stable actuelle de Node.js peut être téléchargée et installée à partir du site Web officiel : <https://nodejs.org/>

Après l'installation, nous vérifions la version à l'aide de la commande : npm -v. Cela affichera les versions installées de Node.js et NPM

##### Création d’un nouveau projet React

Après l'installation réussie de NodeJS et NPM, nous pouvons créer un nouveau projet React en installant temporairement l'outil create-react-app. La création du nouveau projet se fait grâce à la commande : npx create-react-app nomduprojet.

Ici, NPX installera temporairement create-react-app et créera un nouveau projet React nommé ***nomduprojet*** . Notons que ***frontend-events*** est le nom que nous avons choisi pour notre projet React.

Ainsi, l'application que nous avons créée peut s'exécuter localement sur notre système avec les commandes : cd nomduprojet et npm start.

Cela ouvrira l'application React dans un nouvel onglet de notre navigateur avec l'URL : <http://localhost:3000>, le port 3000 étant le port par défaut utilisé par React. Si le port 3000 est occupé par un autre processus, l'application démarrera sur le port 3001 ou sur tout autre port disponible.

A la première exécution de notre application, la page qui s’affiche par défaut est la page :

Figure 17 : Affichage par défaut de notre application React

Une image contenant capture d’écran, Graphique, Police, graphisme

Description générée automatiquement

Nous allons maintenant créer des composants et modifier la page à notre guise.

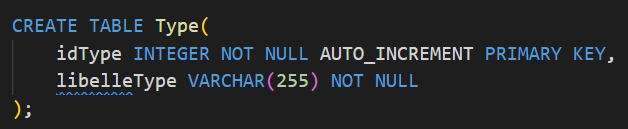
#### III.1.3. La base de données MySQL

Pour créer notre base de données MySQL, nous exécutons les commandes suivantes :

* Script de création de la base de données



* Création de la table Type

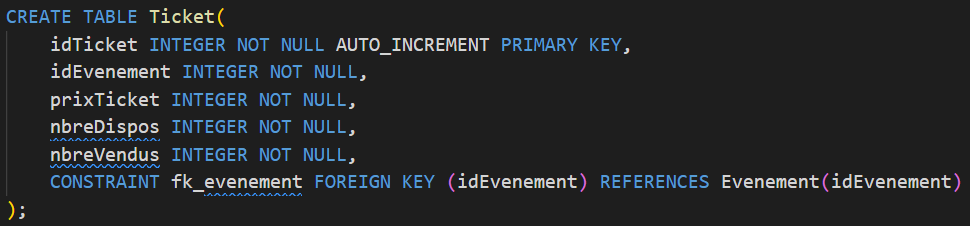


* Création de la table Evènement

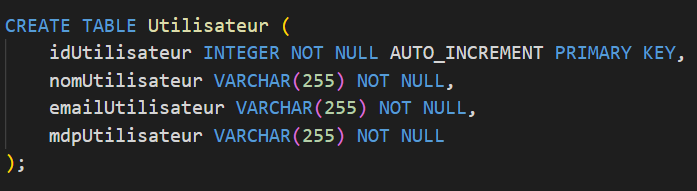
Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

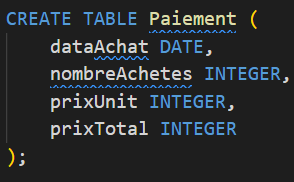
* Création de la table Ticket



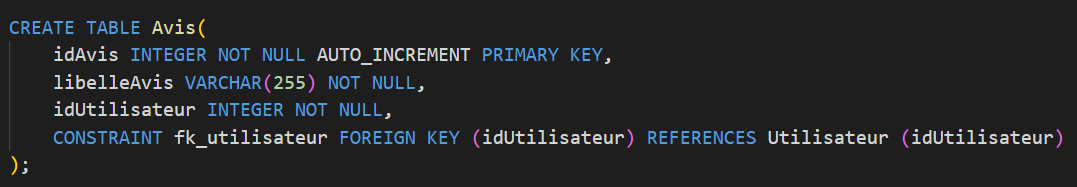
* Création de la table Utilisateur



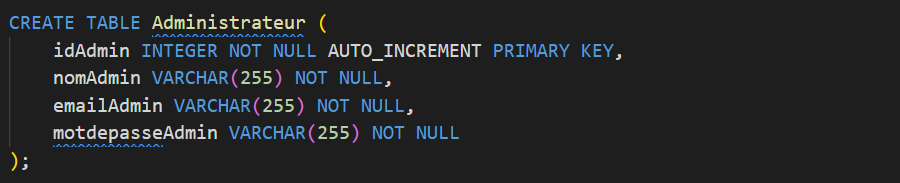
* Création de la table Paiement



* Création de la table Avis



* Création de la table Administrateur



* Création de la table Publicité

Une image contenant texte, Police, Logiciel multimédia, logiciel

Description générée automatiquement

* Création de la table Newsletter

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

## REALISATION

### IV.1. Arborescence globale du projet

#### IV.1.1. Arborescence du backend

Une image contenant texte, capture d’écran, logiciel, multimédia

Description générée automatiquement

Figure 18 : Arborescence du backend

Notre backend est composé d’un dossier principal nommé Backend qui contient le fichier main, le dossier static, le dossier venv et les librairies.

* Le fichier ***main.py*** est le fichier principal de notre application Python. Il contient toutes les fonctions nécessaires au bon déroulement de notre application.
* Le dossier ***static*** est le dossier qui est accessible par le frontend ou toute application publique. Il contient le dossier Images dans lequel sont enregistrées toutes les images de notre projet dès l’exécution de la création sur les tables.
* Le dossier ***venv*** est le dossier qui contient l’environnement de notre application. Il contient l’environnement Python et le SDK.

##### IV.1.2. Arborescence du frontend

##### Arborescence générale du frontend

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 19 : Arborescence principale du frontend

Notre projet contient 4 principaux dossiers et d’autres fichiers. Nous allons donner le contenu des fichiers puis nous allons parcourir ensuite les 4 dossier. Le premier fichier plus important est le fichier App.js.

* **Le fichier App.js** : c’est le fichier principal de notre application. C’est son contenu qui s’affiche à l’exécution de l’application sur <http://localhost:3000>. Après avoir créé nos composants et nos différentes pages, nous devons les importer dans le fichier App.js pour qu’ils s’affichent à l’exécution de notre projet.
* **Le fichier router.js** : c’est le deuxième fichier le plus important ; il contient toutes les routes définies dans notre projet pour la navigation entre les pages.
* **Les fichiers App.css et App.js** : contiennent les configurations Javascript et CSS du fichier principal App.js pour la gestion de la mise en forme et la gestion des évènements.

##### Le dossier node\_modules

Figure 20 : Le dossier node\_modules

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Ce dossier contient tous les packages que nous avons installées et que nous utilisons dans notre projet.

##### Le dossier public

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Figure 21 : Le dossier public

Ce dossier contient les éléments multimédias créés par défaut lors de la création du projet React. Parmi ces éléments, nous avons le logo de React qui s’affiche sur la page d’accueil par défaut et celui qui s’affiche dans le navigateur.

##### Le dossier src

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

##### Figure 22 : Le dossier src

Le dossier src est le dossier dans lequel nous allons organiser notre projet selon nos convenances. Il contient 3 autres dossiers créés par nous :

* Le dossier ***components*** : qui contient les principaux composants que nous importons sur la page App.js.
* Le dossier ***images*** : qui contient les images que nous utilisons dans notre projet.
* Le dossier ***pages*** : il contient les différentes pages que nous avons créées. Il contient, en plus d’autres fichiers, le dossier dashboard qui contient toutes les pages relatives au dashboard.

Le dossier src peut être structuré à la convenance de tout développeur.

### IV.2. Présentation des interfaces

#### IV.2.1. Site principal

Figure 23 : Site principal (partie 1)

Une image contenant texte, bouteille, capture d’écran, nuage

Description générée automatiquement

Le site principal de notre application web se présente comme ci-dessus. Les publicités viennent directement de la base de données et sont récupérées grâce au code suivant :

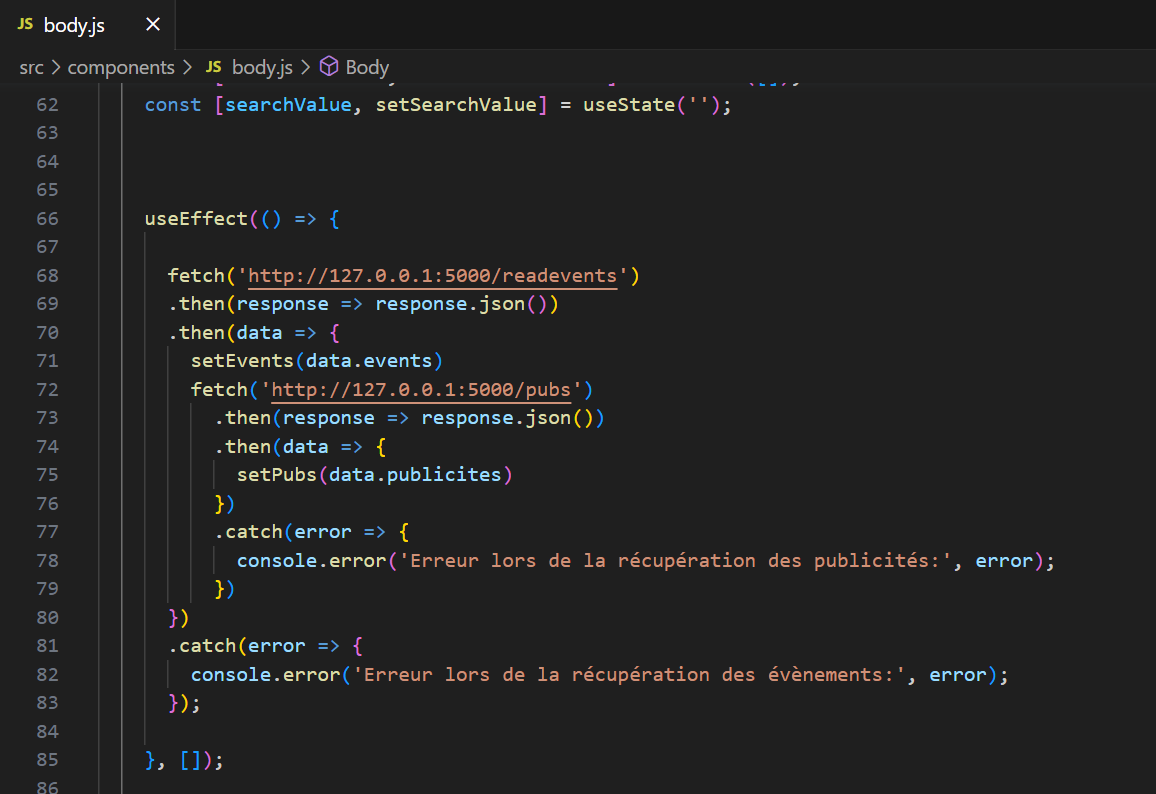


Figure 24 : Fonction Javascript permettant d'afficher la partie 1 du site principal

Figure 25 : Code jsx permettant l'affichage de la partie 1 du site

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 26 : Fonction Python permettant l'affichage des publicités

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 27 : Site principal (partie 2)

Une image contenant texte, capture d’écran, chien, mammifère

Description générée automatiquement

Figure 28 : Site principal (partie 3)

Une image contenant texte, capture d’écran, Site web, Publicité en ligne

Description générée automatiquement

Les évènements affichés sur cette partie de notre site sont ceux de la base de données. Ils sont accessibles grâce aux codes Javascript, Typescript et Python ci-dessous :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Figure 29 : Code Javascript permettant l'affichage des évènements

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure 30 : Code jsx permettant l'affichage des évènements

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure 31 : Code Python permettant l'affichage des évènements

#### IV.2.2. Interface de connexion

Figure 32 : Interface de connexion

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

Figure 33 : Code Javascript permettant la connexion

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure 34 : Code jsx pour le formulaire de connexion

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure 35 : Code Python permettant la connexion

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

La fonction handleLogin() (fig33) est déclenchée par un clic sur le bouton « Connexion » après le remplissage des entrées (fig32). Cette fonction envoie alors une requête fetch à l’API sur le Endpoint ‘/login’ ce qui déclenche l’exécution de la fonction login() (fig35).

#### IV.2.3. Fenêtre d’inscription

Figure 36 : Fenêtre d'inscription

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

Figure 37 : Code Javascript permettant l'inscription

Une image contenant texte, capture d’écran

Description générée automatiquement

Figure 38 : Code jsx pour l'inscription

Une image contenant texte, Appareils électroniques, capture d’écran, affichage

Description générée automatiquement

Figure 39 : Code Python permettant l'inscription

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

La fonction handleRegister() (fig37) est déclenchée par un clic sur le bouton « Inscription » après le remplissage des entrées (fig36). Cette fonction envoie alors une requête fetch à l’API sur le Endpoint ‘/register’ ce qui déclenche l’exécution de la fonction register() (fig39).

#### IV.2.4. Page d’administration

Figure 40 : Page d'administration

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

Figure 41 : Code jsx permettant l'affichage de tous les évènements créés par cet utilisateur

Une image contenant texte, capture d’écran, affichage, logiciel

Description générée automatiquement

Figure 42 : Code Javascript permettant d'afficher les évènements créés par cet utilisateur

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Figure 43 : Code Python permettant d'afficher les évènements créés par l'utilisateur connecté

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

Lorsque la connexion réussit, l’utilisateur est redirigé vers son dashboard. Là, il peut voir tous les évènements, publicités et newsletters qu’il a créés. Le code de la figure 42 envoie une requête fetch à l’API en précisant l’identifiant de l’utilisateur qui est récupéré lors de la connexion. Lorsque l’API reçoit la requête sur la route ‘/readmyevents/id’, la fonction readmyevents() est exécutée et fournit alors la liste des évènements créés par l’utilisateur.

Le même scénario se passe pour l’affichage des concerts, des spectacles, des newsletters, des publicités, etc.

Figure 44 : Affichage de la liste des concerts créés par l'utilisateur connecté

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Figure 45 : Affichage des publicités créées par l'utilisateur connecté

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Figure 46 : Affichage des newsletters créées par l'utilisateur connecté

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

A part l’affichage des ces différentes tables, l’utilisateur peut effectuer plusieurs actions telles que la création et la suppression de nouveaux évènements, de newsletters, de publicités.

Figure 47 : Page de création d'un nouvel évènement

Une image contenant texte, capture d’écran, nombre, logiciel

Description générée automatiquement

Lors de la création d’un nouvel évènement, l’utilisateur soit renseigner toutes les informations nécessaires à l’évènement.

Figure 48 : Fonction Javascript permettant de créer un nouvel évènement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Au clic du bouton « Enregistrer », la fonction handleSubmit() est déclenchée et elle envoie une requête fetch vers l’API.

A l’obtention de la requête fetch, la fonction createvent() est exécutée et les évènements sont créés dans la base de données.

#### Figure 49 : Fonction Python permettant la création d'un évènement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement